Abstract

In this contribution, we present our concepts for two decisive components of a partitioned simulation of fluid-structure interactions: The coupling tool and the flow solver. Hereby, the main focus with respect to the coupling tool, called FSI❄ce, is to achieve a strict separation and, thus, independence of the coupling strategy and the two solvers. This ensures maximal flexibility with respect to the choice or exchange of these components. The flow solvers F3F and Peano are designed with the help of (adaptive) Cartesian grids in combination with space-filling curves such that a maximal storage efficiency both in terms of memory requirements and memory access is reached.

Keywords: Fluid-Structure Interactions, Partitioned Approach, Modularity, Large Deformations, Coupling Environment, Cartesian Grids, Hardware-Efficiency

1 Introduction

Modularity and flexibility are properties which seem to be inherent to partitioned fluid-structure interaction simulations. This is surely true from the conceptual point of view. However, it is not trivial not to loose the independence of the involved components in an actual implementation. Many approaches introduce dependencies between the two solver codes (fluid and structure) by the direct mapping of data between the two (non-matching) solver grids at the interface between fluid and structure. In other cases, the coupling strategy (explicit/implicit coupling,...) is implemented in one of the solvers which unneccessarily complicates the exchange of the respective solver. Our newly developed coupling environment FSI*ce avoids these drawbacks by the introduction of a third, independent mesh describing the fluid-structure interface and by a client-server approach with the two solvers acting as servers and a client controlling the whole coupled simulation and, therewith, of course also defining the coupling strategy

regardless of the actual solvers. This gives us maximal flexibility both in the choice of the two solvers but also in the choice of coupling strategies including sophisticated methods such as multilevel algorithms or reduced order models which are very hard or even impossible to implement without a strict independence of solvers and coupling strategy.

In addition to these requirements, a widely usable software environment for fluid-structure interactions has to be able to efficiently handle large deformations of the computational domain. This holds in particular for the fluid domain. Think of particles moving through the fluid, e.g. To handle this requirement, we present a fluid solver working on (adaptive) Cartesian grids in an Eulerian framework that is with a fixed grid. We show methods to fastly update the grid after a change of the geometry, to efficiently map data from these Cartesian grids to the central interface mesh of FSI*ce as well as robust interpolation schemes tailored to Cartesian grids and preventing instabilities.

Finally, we will present results achieved with our Cartesian flow solvers and FSI*ce for benchmark and application scenarios in various configurations.

2 FSI❄ce: Modular Coupling of Codes

2.1 Basic Concept of the Coupling Environment FSI❄ce

The aim of the coupling environment FSI❄ce as proposed in [5] is to enable a partitioned simulation of fluid-structure interactions maintaining the modularity and flexibility of the theoretical concept of partitioned simulations in a real implementation. Thus, we have to ensure certain properties: To reduce the cost of embedding a solver in the coupled simulation environment, the solvers have to act purely as solvers and not additionally as a steering component of the coupled simulation, for example. To maintain the independence of the fluid and the structure solver, the two solvers may not communicate directly with each other. To avoid dependencies between solvers and the logically third component of a coupled simulation, the coupling strategy, it has to be implemented in a own software component and not in one or both of the solvers. As the simulation of fluid-structure interactions is a typical high performance computing application, the coupling tool has to be able to handle parallel solvers.

According to these requirements, FSI❄ce acts not only as a communication interface between the two solvers involved but as a supervisor of the coupled simulation (see Fig. 1). The coupling strategy is chosen and executed within FSI❄ce. For this purpose, FSI❄ce evaluates the simulation results recieved from the two solvers and decides what to do in the next step. This decision is sent to the solvers together with updated function values at the coupling surface. Hereby, three possible responses are provided: `FSII_STATUS_GO` (continue with the next time step), `FSII_STATUS_LOOP` (perform another loop in an implicit coupling scheme), and `FSII_STATUS_STOP` (finalise the computation).

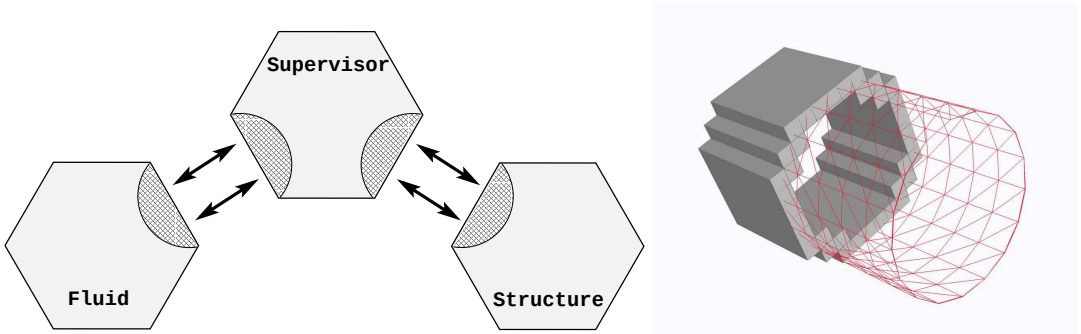


Figure 1: Left: Schematic view of the concept of FSI❄ce with FSI❄ce acting as a supervisor of the whole coupled simulation (from [4]); right: Cells at the boundary of the computational domain of a fluid solver using a Cartesian grid together with the central surface description of a cylinder surface held in FSI❄ce.

A so-called central surface mesh, that is a triangulation of the coupling surface between fluid and structure, is held within FSI❄ce as a reference surface description for the two solvers. That is, the solvers map their data to this central mesh or get updated informations from this central mesh. Thus, they only send or recieve data to or from FSI❄ce and not from the second solver. Fig. 1 shows a Cartesian solver grid together with the central surface mesh. The central surface mesh is realised as a vef(vertex-edge-face)-graph and has to describe a closed and oriented surface without self intersections. Besides the pure surface description, the vef-graph also holds data stored at the vertices such as velocity and force values. Each of the involved solver codes has to hold his own instance of the central surface mesh in order to be able to send data in the correct form and to interpret recieved data in an appropriate manner. For the initialisation and updating of the central surface mesh, FSI❄ce provides two possibilities: The mesh can either be identical to the surface mesh of one of the solver grids or defined completely separately.

The programming interface of FSI❄ce was first described in [5]. In this early publication, only the communication via MPI (Message Passing Interface [33]) was proposed. However, if we want to use parallel fluid and structure solvers (which is inevitable due to the complexity of most application scenarios), these solvers themselves use MPI as a communication library. This poses some new problems in the usage of the MPI Communicator `MPI_COMM_WORLD`. The usage of collective MPI functions in the fluid solvers, for example, would require respective function calls also in the structure solver where they are in general not available. As it does not make sense to exculde collective MPI functions, FSI❄ce implements two possible solutions to this problem:

The first possibility is that each solver and FSI❄ce have to introduce their own communicators (`FLUID_COMM`, `STRUC_COMM`, and `FSI_COMM`) which do not have common processes in `MPI_COMM_WORLD`. This requires only slight modifications of the MPI calls within the two solvers.

In the second possibility, we leave the solver implementations unchanged and use direct communication via sockets (TCP/IP) for the communication between FSI❄ce

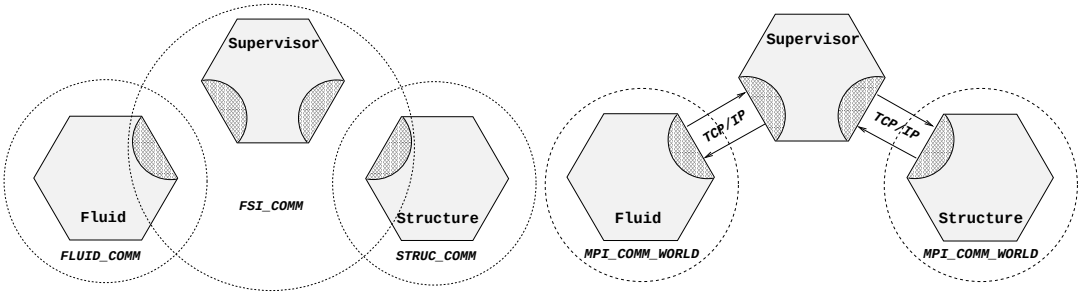


Figure 2: Schematic view of the two communication concepts provided by FSI❄ce: Using newly defined MPI communicators for each of the involved three software components (left); communication between solvers and FSI❄ce via sockets (right).

```
while (status == FSII_STATUS_GO)
  do
    compute the current time step;
    write data to DMesh;
    status = Fsi_send_quantity(DMesh,...);
    status = Fsi_recieve_quantity(DMesh,...);
    read data from DMesh;
  until (status != FSI_STATUS_LOOP)
  t = t + dt;
end
```

Figure 3: Modified time loop of the fluid or structure solver for a coupling with FSI❄ce (single level coupling).

and the solvers. Besides the advantages that it requires no changes in the solver codes, this concept is also more widely applicable. Whereas 1) requires a homogeneous implementation of MPI on all involved computing resources, the communication via sockets can be realised also in very heterogeneous environments. Such, running the whole coupled simulation in a grid environment with FSI❄ce running on a PC and the two solver running on different high performance computers, e.g., becomes possible. Both concepts are displayed schematically in Fig. 2 and explained in more details in [4].

2.2 Potentials for Sophisticated Coupling Schemes

2.2.1 Methods with Explicit Subiterations Within one Time Step

Due to the strict separation of the solver codes from the coupling strategy realised in FSI❄ce, the work to be done within the solvers to tie them to FSI❄ce is very small. As a simple example, Fig. 3 shows the modified solver time loop needed to perform a single level coupling with possible subiterations within each time step. In particular, the algorithm shows, that the decision on subiterations, their convergence, and the computation of Aitken relaxation factors [35], e.g., is not performed in the solver but within FSI❄ce where the value of the variable `status` is set.

```
while (status == FSII_STATUS_GO)
   do
      compute the current time step;
      write data to DMesh;
      status = Fsi_send_quantity(DMesh,...);
      status = Fsi_recieve_quantity(DMesh,...);
      read data from DMesh;
      if (status == FSII_STATUS_RESTRICT)
         restrict to coarser grid;
      else if (status == FSI_STATUS_INTERPOLATE)
         interpolate to finer grid;
   until (status == FSI_STATUS_GO || status == FSI_STATUS_STOP)
   t = t + dt;
end
```

Figure 4: Modified time loop of the fluid or structure solver for a coupling with FSI❄ce (single or multilevel coupling).

2.2.2 Multilevel Methods

To implement multilevel coupling schemes such as those presented in [3, 51], additional values for the status variable indicating whether the solver has to switch to another grid level have to be introduced. But still, we end up with a single modification of the time loop of the solvers as displayed examplarily in Fig. 4. Note that this algorithm is a generalisation of the single level algorithm and can be applied for both single and multilevel coupling schemes.

2.2.3 Implicit Coupling Iterations

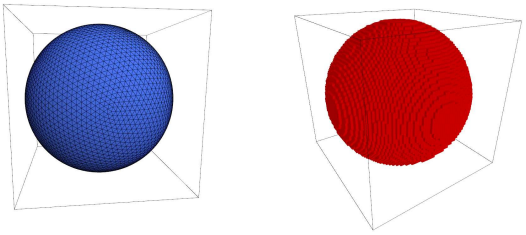
For some problems, explicit coupling iterations do not lead to a satisfying overall convergence. Thus, implicit coupling iterations have to be considered as well. Hereby, in general, Jacobians of the structure and the fluid solver are required, the computation of which, however, is not possible outside the solver codes. Thus, the implementation of implicit coupling iterations seems to contradict the FSI❄ce paradigma of performing the whole coupling independent from the solver codes. However, the usage of for example reduced order models [48] purely relying on information gained as a response of the solvers at the coupling interface, can be done in a very natural way within FSI❄ce. Fig. 5 shows the overall iteration loop with the assignment of the single steps to FSI❄ce or the solvers. It becomes obvious, that we can stay with the time loop displayed in Fig. 3 within the solver codes.

2.3 Octree-Based Data Access

As we typically deal with three different and non-matching grids – a fluid solver grid, a structure solver grid, and the central surface mesh – in our fluid-structure environment, efficient methods for the mapping of data between these grids at the coupling surface are of utmost importance. Although the actual choice and, in some special cases, also the implementation of the respective interpolations or projections is left to the user,

```
do
    build reduced order model structure;
    apply reduced order model structure;
    call fluid solver;
    build reduced order model fluid;
    apply reduced order model fluid;
    call structure solver;
until convergence
```

Figure 5: Iteration loop for an implicit coupling using reduced order models both for the structure and the fluid solver [48] performed by FSI❄ce. Steps that are performed by the solvers are written in italics.



Cart. res.	# triang.	runtime
256	8,000	2.6 sec
512	8,000	10.1 sec
512	32,000	14.3 sec
512	128,000	17.4 sec

Figure 6: Visualisation of a surface triangulation (left) and the Cartesian solver grid (middle) for a three-dimensional spherical computational domain; right: Runtime for the neighbourhood search between vertices of the Cartesian solver grid and the triangles of the central surface mesh measured on a Pentium M 1.6 GHz processor with 2048 kB cache (from [6]).

a user friendly coupling environment has to provide functions for the technical and general part of the data mapping – the detecting of nearest neighbours of vertices and elements of the involved grids. For this purpose, FSI❄ce embeds the vef-graph of the central surface mesh in an adaptive octree structure. This octree allows for a very efficient and, thus, fast detection of the nearest grid vertices of a solver grid for each surface triangle. Hereby, the keys to efficiency are the adaptive structure of the octree with fine cells only around the coupling surface and not in the whole computational domain and, second, the exploitation of information inheritance from father to son cells [14, 8]. To give a first impression of the efficiency of the neighbourhood search, Fig. 6 shows computational times for the neighbourhood search between a Cartesian solver grid and the surface mesh of a sphere in three dimensions.

3 F3F and Peano: Efficient Flow Simulation on Cartesian Grids

Our flow solvers F3F and Peano both solve the Navier-Stokes equations

$$\frac{\partial u}{\partial t} + \frac{1}{re}\Delta u + (u \cdot \nabla)u + \nabla p = 0 \quad (1)$$

$$\nabla^T u = 0 \quad (2)$$

	bytes per vertex	bytes per cell
2D	20	14
3D	28	18

Table 1: Overall storage requirements per grid vertex (where flow velocities are stored) and grid cell (where pressure values are stored) for the Peano flow solver using DaStGen [10, 11].

with flow velocities u, pressure p, and Reynolds number re using an explicit time-stepping scheme in combination with a Chorin-like projection method [12, 19, 7]. For the spatial discretisation, F3F uses a finite volume discretisation [5, 8] whereas Peano works with a finite element discretisation [50, 37, 8, 7].

3.1 Benefits of Cartesian Grids

Our flow solvers F3F and Peano are based on Cartesian computational grids, which showed to be a competitive choice in the last years [1, 42, 21, 23, 15, 2]. As such grids poccess a clearly defined structure in the regular but also in the adaptive case, this choice minimises the storage requirements. We do not have to store any interdependencies between grid components such as vertices, edges, faces and elements. They are inherently given by the locally recursive construction or refinement process of the grids. Our adaptive solver Peano uses in addition a cell-wise operator evaluation, which makes also the storage of specialised difference stencils at boundaries between different refinement depths obsolete as the cell-wise operators only rely on vertex and midpoint data of the current cell. These informations are available equally in all cells independent of the refinement depth of neighbouring cells. To give consideration to adaptive grid refinement, we only have to correctly restrict and interpolate operator values or data, respectively, at such boundaries between different mesh widths [21, 41, 23, 7]. To further minimise the storage requirements, we implemented a data structure generator DaStGen [10], which prevents the useless storage of temporarily not required informations, e.g. Table 1 shows the resulting storage requirements per vertex (where we store the flow velocities, e.g.) and per grid cell (where we store the pressure, e.g.). Note that these number do not depend on the adaptivity pattern of the grid and that the storage requirements have been reduced by a factor of approximately four in the two-dimensional case compared to the requirements given in [7] without the usage of DaStGen.

Whereas we can apply a simple i, j, k-indexing and processing order of the grid cells combined with a simple partitioning of the domain in rectangular subdomains for parallelisation [36], the algorithm becomes more complex in Peano due to the grid adaptivity. Here, we have to be very careful to maintain a high efficiency of data storage and data access in spite of the less regular data dependencies. This holds in particular in the case of multilevel algorithms, where also data dependencies between different refinement levels have to be taken into consideration. In addition, we will not achieve a balanced parallelisation using rectangular subdomains any more. And

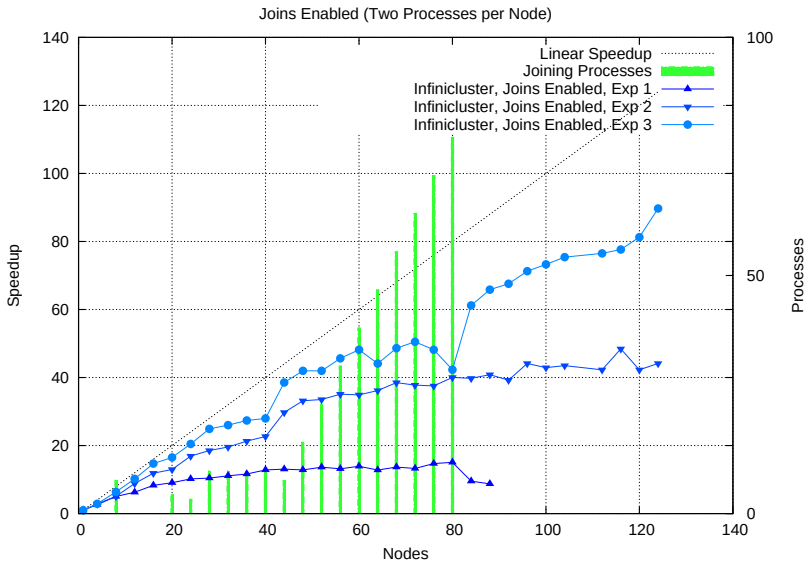


Figure 7: Preliminary results on the strong scaling of the Peano solver for a Poisson equation solved on an InfiniBand cluster. Experiment 1 includes $6.1 \cdot 10^5$ degrees of freedom, experiment 2 $5.4 \cdot 10^6$ degrees of freedom, and experiment 3 $4.9 \cdot 10^7$ degrees of freedom.

we need a dynamical load balancing as a consequence of dynamic adaptivity. Peano solves all these difficulties with the help of space-fiiling curves [44], in particular Peano curves. For each of our adaptively refined Cartesian grids, there is a corresponding iterate of the Peano curve prescribing an ordering of the grid cells on *all* refinement levels. Due to the good locality properties of these curves [52], we use them to define the processing order of our grid cells within the solver iterations. On this basis, we construct highly cache-efficient data structures, so-called stacks [21, 41] for the storage of the degrees of freedom within these iterations. Such, we achieve L2 cache-hitrates above 99% or, in other words and more significant, only 110% of the unavoidable number of L2 cache misses [21, 41, 50, 32]. Note that this is true not only for single level solvers but also for multigrid methods [32, 28].

For the parallelisation of solvers on adaptive grids, space-filling curves are known to be an efficient and comparably easy to apply tool for domain partitioning and dynamical load balancing [38, 39, 20, 43, 40, 30, 52, 45]. Hereby, the principle idea behind is to queue up the grid cells (on all refinement levels) in the order prescribed by the space-filling curve and, subsequently, to cut this queue into equal pieces. Peano adopts this principle [29, 24, 22]. In the last two years, the original concept has been modified to a more level-oriented approach which has the advantage that already the domain partitioning itself can be done in parallel [36, 7, 31]. Thus, we never have to store the whole grid on one single processor. Fig. 7 shows some preliminary results on the scalability of the code. More details of the new domain decomposition and load balancing approach will be published soon in a separate paper.

From the numerical point of view, Peano offers a highly flexible and powerfull

grid adaptivity [16, 28, 32] in combination with an efficient multigrid solver [28, 32]. To remedy the lower discretisation order at complicated domain boundaries induced by the Cartesian grid, a variety of possibilities is known in literature [49, 34]. The implementation of such methods in Peano is work in progress.

3.2 Consistent Forces

For the simulation of fluid-structure interaction problems, an obviously very important feature of a flow solver is the accurate calculation of forces exerted on the structure surface by the flow field. Hereby, the method of consistent forces as proposed in [18] can be applied very naturally in our solvers. The local forces at grid vertices at the structure boundary are computed by the accumulation of the contributions to the right hand side of the momentum equation (1) from all adjacent fluid cells (see also [7]):

$$f_i = \left(A\frac{du_h}{dt} + Du_h + C(u_h)u_h - M^T p_h \right)_i, \tag{3}$$

where A, D, C, and M denote the discrete mass, diffusion, convection, and gradient operators, respectively. In the adaptive solver Peano, these values are automatically available after each grid-wide cell-wise operator evaluation. More details for the realisation in our solves are provided in [9, 7, 5, 8].

3.3 Geometry Changes

Due to structure movements or deformation, the computational domain of the fluid solver is underlying possibly large deformations or even topology changes. This requires an adequate method to handle these changes. Our flow solvers work with the Eulerian approach that is with fixed grids. These grids cover a rectangular domain in which the actual computational domain is embedded. Grid cells outside the considered scenario and grid cells within the structure are marked as 'obstacle cells'. This approach is called a marker-and-cell approach [46, 19] in literature. For regular grids, a geomatry change simply results in a reset of the cell markers. It is obvious that the upper bound of the costs for this change corresponds to one sweep over all grid cells. In the case of adaptively refined grids such as those displayed in Fig. 8, adaptive refinements, in particular those around complicatetly shaped structures have to be redone in addition. However, these costs are proportional to the costs of one sweep over the grid, again, as even the complete generation of a new grid from the current central surface mesh requires only one sweep over the grid which is being build immediately within this sweep [14]. In [14, 7, 6], we give runtimes for this grid generation for different grid resolutions. For example, the generation of a grid with 843 million grid cells requires only a computational time of eleven second on a Pentium 4 2.4 Ghz processor [6], which is really neglectable compared to the runtime of a fluid solver time step on such a fine grid. If we implement the restriction of changes to those subtrees of the cell tree which is actually affected by a geometry movement – which

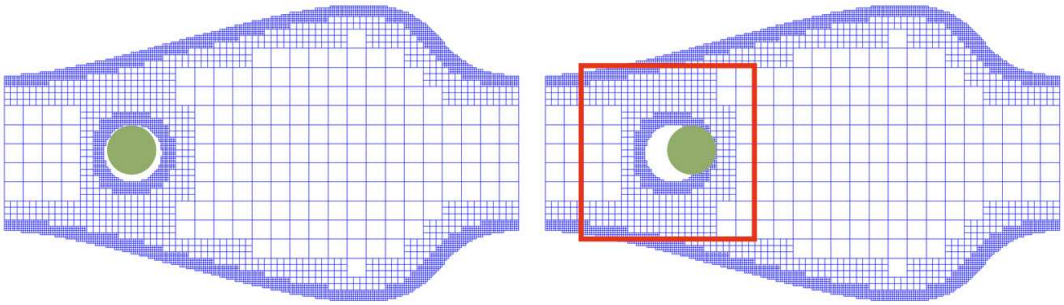


Figure 8: Left: Adaptively refined Cartesian grid for the simulation of the transport of a spherical particle in a channel with oscillating diameter. Only fluid cells are displayed, regions with obstacle cells appear white or green, respectively. Right: Situation after a particle movement. The subtree of the grid cell tree where changes have to be performed is marked with a red square.

is currently work in progress, runtimes for resetting the grid after a geometry change will be further reduced.

The change of cell markers and adaptive refinement patterns due to geometry changes leads to new fluid grid points for which correct velocity values have to be determined that do not harm the cell-wise continuity equations. To find such values, we use the values from the nearest already existing fluid grid points and subsequently apply an L_2-projection of the velocity field onto the closest divergence-free velocity field [7]: $\vec{u}^{new} = \vec{u}^{old} - \nabla q$ with $\Delta q = \nabla^T \vec{u}^{old}$.

3.4 Surface Data Mapping

To map data between the central surface mesh of FSI❄ce and the Cartesian grid of the flow solver, we first determine the nearest surface triangle for each boundary vertex of the Cartesian solver grid as described in Sect. 2.3. Subsequently, we determine the orthogonal projection from the vertex to the respective triangle.

3.4.1 Velocities

To transport information from the structure solver to the fluid solver, we use the velocities at the surface of the structure computed by the structure solver. The velocities at the Cartesian boundary vertices are set equal to the values at the projection points at the nearest triangle of the central surface mesh. Hereby, we interpolate the values linearly within the triangles [13].

3.4.2 Forces

Forces exerted on the structure are the input the structure solver gets from the fluid solver. Thus, we have to perform the invers mapping as for the velocities: We set

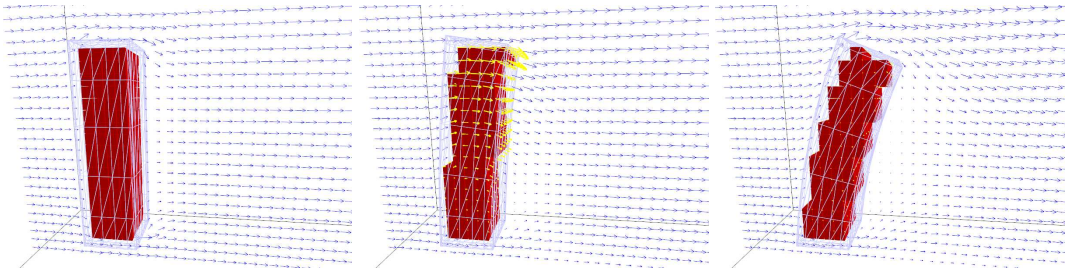


Figure 9: Snapshots from simulations of fluid flow around a tower in a laminar flow regime [13].

values at projection points of Cartesian boundary points equal to the forces computed at these boundary vertices. Subsequently, these values are distributed to the nodes of the central surface mesh according to the barycentric coordinates of the projection points.

As the structure solver we use in our examples, AdHoc [17], needs stresses and not forces as an input, we additionally compute the stresses at the central surface mesh. For this purpose, we define polygonals around the nodes of the surface mesh defined the baycentres of the neighbouring triangles and the midpoints of the involved edges. The stresses at the surface nodes result from the forces devided by the area of these polygonals [13].

4 Numerical Results

In this section, we propose some scenarios for which we performed simulations using our coupling environment FSI❄ce in combination with our flow solver F3F and Peano. Currently, these simulations can be devided into two subgroups, test cases and benchmarks with the purpose of an evaluation and verification of our software and application scenarios with the task of gaining new insights in the underlying phyiscal phenomena. We will not go into details for the examples shown but, instead, only give an overview of some scenarios which have already been computed with our tools.

4.1 Flow Around a Tower

A first test case to show the technical and quantitative function of our software components is the flow around a tower, for which several simulations have been run in [13, 6]. To simulate the structure behaviour, we used the h-p-adaptive structure solver AdHoc [17]. Fig. 9 shows snapshots from simulations runs presented in [13].

4.2 Benchmark Scenarios

In [25], several benchmark scenarios for the evaluation and verification of simulation tools for fluid-structure interactions have been proposed. We are currently performing

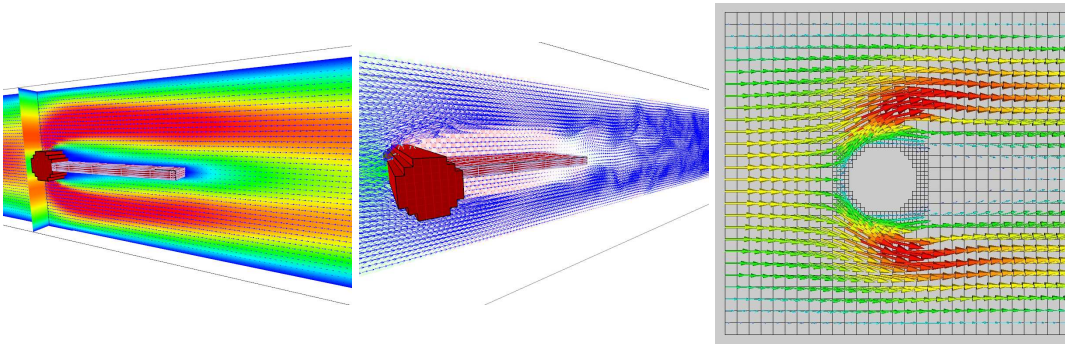


Figure 10: Left and middle: First simulation results for the three CFD benchmark scenarios described in [25]. Right: Cut-off of the flow field for the CFD benchmark 'flow around a cylinder' with Reynolds number 20 from [47] (see also [6]).

first computations for the FSI benchmarks I-III. As we can not present the results in this paper, yet, we give some pictures and data gained from preliminary studies of pure CFD benchmarks which were performed in preparation of the whole coupled fluid-structure simulations. Fig. 10 shows visualisations of the CFD scenarios from [25] and gives a cut-off of the flow field around a cylinder computed on an adaptively refined grid. For a grid with $42,501$ cells, we achieved a drag coefficient of 5.691 (reference value 5.580) and a lift coefficient of 0.0113 (reference value 0.0107) [7]. These results will be further improved in the future by a more sophisticated and dynamical grid adaptivity.

4.3 The Drift Ratchet

The Drift Ratchet is our application scanario. A drift ratchet is a tool used in life sciences for the separation of microscopic particles according to their size. The function of this tool has been shown experimentally [27]. However, a deeper understanding of the crucial physical phenomena and an exact tuning of the concrete realisation require further investigations, among which numerical simulations are a very meaningful and fast possibility. The drift ratchet consists of a kind of sieve with pores with asymmetrically oscillating diameter [7, 27]. A fluid with the suspended particles is pumped forward and backward through this sieve by a pressure pump with oscillating pressure. In the last months, we performed several simulation runs for the transport of one particle in a suitable cut-off of a pore. First 'drift' effects could be shown and are described in [7, 6]. Here, we only show some snapshots of a simulation run for illustration in Fig. 11. The particles are rigid bodies for which we only have to compute velocities and rotation according to Newtons laws of motion [7]. This results in very simple structure equations which we directly solve within our fluid solver.

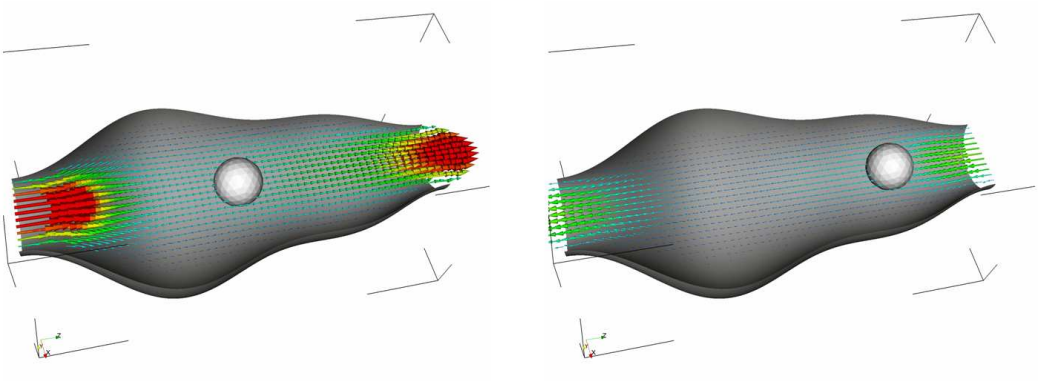


Figure 11: Snapsots of a simulation of a particle moving in a cut-off of a drift ratchet pore.

5 Conclusion

In this paper, we have described and showed the applicability of our software components for the partitioned fluid-structure interaction simulations: The coupling environment FSI❄ce and the flow solvers F3F and Peano. For the coupling environment, we could show that we achieve a very high flexibility in the choice and exchange of solvers and the coupling strategy which is not the case for example for the most commonly used tool for code to code coupling MpCCI [26, 8, 7], for example. The flow solvers show the storage efficiency of Cartesian grids in particular in combination with the newly developed algorithms and data structures based on space-filling curves. In addition, large geometry or even topology changes can be realised very efficiently within our Eulerian approach as grid generation and adaption are very fast processes within the framework of our Cartesian grids. In some first examples, the technical and quantitative correctness and efficiency has been shown. However, exploiting the whole potential of the concept is still work in progress. This includes the implementation of more sophisticated coupling strategies, libraries for data interpolation and projection between solver grids and the central surface description of FSI❄ce as well as further runtime optimisations, better time stepping methods, and dynamical adaptivity for our flow solver Peano.

Acknowledgements

This work was supported by the German Research Foundation, project HA 1517/25 and research group 493, and the Competence Network for Technical, Scientific High Performance Computing in Bavaria. This support is thankfully acknowledged.

References

[1] M. Bader, H.-J. Bungartz, A. Frank, and R.-P. Mundani. Space tree structures for PDE software. In P. M. A. Sloot, C. J. Kenneth Tan, J. J. Dongarra, and A. G. Hoekstra, editors, *Proc. of the Int. Conf. on Comp. Science*, number 2331 in LNCS, pages 662–671. Springer, 2002.

[2] J. Ballmann, K.-H. Brakhage, F. Bramkamp, W. Dahmen, B. Gottschlich-Müller, M. Hesse, Ph. Lamby, and S. Müller. Polyhedral discretization, data compression and mesh generation. *Numerical Notes on Fluid Mechanics*, 84:125–204, 2003.

[3] H. Bijl, A. H. van Zuijlen, and S. Bosscher. Two level algorithms for partitioned fluid-structure interaction computations. In P. Wesseling, E. Oñate, and J. Périaux, editors, *ECCOMAS CFD 2006, European Conference on Computational Fluid Dynamics*. TU Delft, 2006.

[4] M. Brenk. *Algorthmic Aspects of Fluid-Structure Interactions on Cartesian Grids (German: Algorithmische Aspekte der Fluid-Struktur-Wechselwirkung auf kartesischen Gittern)*. PhD thesis, TU München, 2007.

[5] M. Brenk, H.-J. Bungartz, M. Mehl, R.-P. Mundani, A. Düster, and D. Scholz. Efficient interface treatment for fluid-structure interaction on cartesian grids. In *ECCOMAS COUPLED PROBLEMS 2005, Proc. of the Thematic Conf. on Computational Methods for Coupled Problems in Science and Engineering*. International Center for Numerical Methods in Engineering (CIMNE), 2005.

[6] M. Brenk, H.-J. Bungartz, M. Mehl, I.L. Muntean, T. Neckel, and K. Daubner. An eulerian approach for partitioned fluid-structure simulations on cartesian grids. *Computational Mechanics*, 2008. accepted.

[7] M. Brenk, H.-J. Bungartz, M. Mehl, I.L. Muntean, T. Neckel, and T. Weinzierl. Numerical simulation of particle transport in a drift ratchet. *SIAM Journal of Scientific Computing*, 2008. accepted.

[8] M. Brenk, H.-J. Bungartz, M. Mehl, and T. Neckel. Fluid-structure interaction on cartesian grids: Flow simulation and coupling environment. In H.-J. Bungartz and M. Schäfer, editors, *Fluid-Structure Interaction*, number 53 in LNCSE, pages 233–269. Springer, 2006.

[9] M. Brenk, H.-J. Bungartz, and T. Neckel. Cartesian discretisations for fluid-structure interaction – consistent forces. In P. Wesseling, E. Oñate, and J. Périaux, editors, *ECCOMAS CFD 2006, European Conference on Computational Fluid Dynamics*. TU Delft, 2006.

[10] H.-J. Bungartz, W. Eckhardt, M. Mehl, and T. Weinzierl. Dastgen - a data structure generator for parallel c++ hpc software. In Bubak, van Albada, Sloot, and Dongarra, editors, *ICCS 2008 Proceedings*, Lecture Notes in Computer Science, Heidelberg, Berlin, June 2008. Springer-Verlag.

[11] H.-J. Bungartz, M. Mehl, and Ch. Zenger. *100 Volumes NNFM and 40 Years Numerical Fluid Mechanics*, volume 100 of *Notes on Numerical Fluid Mechanics and Multidisciplinary Design*, chapter Computer Science. Springer, 2008.

[12] A. J. Chorin. Numerical solution of the Navier-Stokes equations. *Math. Comp.*, 22:745–762, 1968.

[13] K. Daubner. Data exchange and geometry treatment for the simulation of fluid-structure interactions with partitioned aproaches (german: Datenaustausch und geometriebehandlung bei der simulation von fluid-struktur-wechselwirkungen mit partitionierten ansätzen). Diploma thesis, Institut für Informatik, TU München, 2005.

[14] K. Daubner. Geometrische Modellierung mittels Oktalbäumen und Visualisierung von Simulationsdaten aus der Strömungsmechanik. Studienarbeit, Universität Stuttgart, Universität Stuttgart, 2005.

[15] F.J. Deister. *Selbstorganisierendes hybrid-kartesisches Netzverfahren zur Berechnung von Strömungen um komplexe KOnfigurationen*. PhD thesis, Universität Stuttgart, 2002.

[16] N. Dieminger. Kriterien für die selbstadaption cache-effizienter mehrgitteralgorithmen. Diplomarbeit, Fakultät für Informatik, Technische Universität München, 2005.

[17] A. Düster, H. Bröker, H. Heidkamp, U. Heißerer, S. Kollmannsberger, R. Krause, A.Muthler, A. Niggl, V. Nübel, M. Rücker, and D. Scholz. *AdhoC^4 – User's Guide*. Lehrstuhl für Bauinformatik, TU München, 2004.

[18] P. M. Gresho and R. L. Sani. On pressure boundary conditions for the incompressible Navier-Stokes equations. *Int. J. Numer. Meth. Fluids*, 7:1111–1145, 1987.

[19] M. Griebel, Th. Dornseifer, and T. Neunhoeffer. *Numerical Simulation in Fluid Dynamics, a Practical Introduction*. SIAM, 1998.

[20] M. Griebel and G. W. Zumbusch. Hash-storage techniques for adaptive multilevel solvers and their domain decomposition parallelization. In J. Mandel, C. Farhat, and X.-C. Cai, editors, *Proceedings of Domain Decomposition Methods 10, DD10*, number 218, pages 279–286, Providence, 1998. AMS.

[21] F. Günther. *Eine cache-optimale Implementierung der Finiten-Elemente-Methode*. PhD thesis, Institut für Informatik, TU München, 2004.

[22] F. Günther, A. Krahnke, M. Langlotz, M. Mehl, M. Pögl, and Ch. Zenger. On the parallelization of a cache-optimal iterative solver for pdes based on hierarchical data structures and space-filling curves. In *Recent Advances in Parallel Virtual Machine and Message Passing Interface: 11th European PVM/MPI Users Group Meeting Budapest, Hungary, September 19 - 22, 2004. Proceedings*, volume 3241 of *Lecture Notes in Computer Science*, Heidelberg, 2004. Springer.

[23] F. Günther, M. Mehl, M. Pögl, and C. Zenger. A cache-aware algorithm for PDEs on hierarchical data structures based on space-filling curves. *SIAM J. Sci. Comput.*, 28(5):1634–1650, 2006.

[24] W. Herder. Lastverteilung und parallelisierte Erzeugung von Eingabedaten für ein paralleles cache-optimales Finite-Element-Verfahren. Diploma thesis, Institut für Informatik, TU München, 2005.

[25] J. Hron and S. Turek. Proposal for numerical benchmarking of fluid-structure interaction between elastic object and laminar incompressible flow. In H.-J. Bungartz and M. Schäfer, editors, *Fluid-Structure Interaction*, number 53 in LNCSE, pages 371–385. Springer, 2006.

[26] W. Joppich, M. Kürschner, and the MpCCI team. Mpcci — a tool for the simulation of coupled applications. *Concurrency Computat.: Pract. Exper.*, 18(2):183–192, 2006.

[27] C. Kettner, P. Reimann, P. Hänggi, and F. Müller. Drift ratchet. *Phys. Rev. E*, 61:312–323, 2000.

[28] A. Krahnke. *Adaptive Verfahren höherer Ordnung auf cache-optimalen Datenstrukturen für dreidimensionale Probleme.* Dissertation, TU München, 2005.

[29] M. Langlotz. Parallelisierung eines cache-optimalen 3d finite-element-verfahrens. Diplomarbeit, Fakultät für Informatik, Technische Universität München, 2004.

[30] A. Laszloffy, J. Long, and A. K. Patra. Simple data management, scheduling and solution strategies for managing the irregularities in parallel adaptive hp finite element simulations. *Parallel Computing*, 26, 2000.

[31] M. Mehl, M. Brenk, I.L. Muntean, T. Neckel, and T. Weinzierl. Benefits of structured cartesian gris for the simulation of fluid-structure interactions. In *Proceedings of the Third Asian-Pacific Congress on Computational Mechanics*, Kyoto, Japan, December 2007.

[32] M. Mehl, T. Weinzierl, and C. Zenger. A cache-oblivious self-adaptive full multigrid method. *Numer. Linear Algebr.*, 13(2-3):275–291, 2006.

[33] MPI: A message-passing interface standard, version 1.1, 1995. Manual.

[34] R. Mittal and G. Iaccarino. Immersed boundary methods. *Annu. Rev. Fluid Mech.*, 37:239–261, 2005.

[35] D. P. Mok and W. A. Wall. Partitioned analysis schemes for the transient interaction of incompressible flows and nonlinear flexible structures. In W. A. Wall, K.-U. Bletzinger, and K. Schweizerhof, editors, *Trends in Computational Structural Mechanics*, pages 689–698. CIMNE, 2001.

[36] Ioan Lucian Muntean, Miriam Mehl, Tobias Neckel, and Tobias Weinzierl. *High Performance Computing in Science and Engineering, Garching 2007*, chapter Concepts for Efficient Flow Solvers Based on Adaptive Cartesian Grids. Springer, Berlin Heidelberg New York, 2008.

[37] T. Neckel. Einfache 2d-Fluid-Struktur-Wechselwirkungen mit einer cache-optimalen Finite-Element-Methode. Diploma thesis, Fakultät für Mathematik, TU München, 2005.

[38] J.T. Oden, A. Para, and Y. Feng. Domain decomposition for adaptive *hp* finite element methods. In *Domain decomposition methods in scientific and engineering computing. Proceedings of the 7th international conference on domain decomposition*, volume 180 of *Contemp. Math.*, pages 203–214, Providence, Rhode Island, 1994. AMS.

[39] M. Parashar, J.C.Browne, C.Edwards, and K.Klimkowski. A common data management infrastructure for parallel adaptive algorithms for pde solutions. In *Proceedings of the 1997 ACM/IEEE conference on Supercomputing*, pages 1–22. ACM Press, 1997.

[40] A.K. Patra, J. Long, and A. Laszloff. Efficient parallel adaptive finite element methods using self-scheduling data and computations. *HiPC*, pages 359–363, 1999.

[41] M. Pögl. *Entwicklung eines cache-optimalen 3D Finite-Element-Verfahrens für große Probleme*, volume 745 of *Fortschritt-Berichte VDI, Informatik Kommunikation 10*. VDI Verlag, Düsseldorf, 2004.

[42] T. Pohl, M. Kowarschik, J. Wilke, K. Iglberger, and U. Rde. Optimization and profiling of the cache performance of parallel lattice boltzmann codes. *Parallel Processing Letters*, 13:549–560, 2003.

[43] S. Roberts, S. Klyanasundaram, M. Cardew-Hall, and W. Clarke. A key based parallel adaptive refinement technique for finite element methods. In *Proc. Computational Techniques and Applications: CTAC '97*, pages 577–584, Singapore, 1998.

[44] H. Sagan. *Space-filling curves*. Springer, New York, 1994.

[45] J. Steensland, S.C. Handra, and M. Parashar. An application-centric characterization of domain-based sfc partitioners for parallel samr. *IEEE Trans. on Parallel and Distributed Systems*, 13(12):1275–1289, 2002.

[46] M. F. Tomé and S. McKee. GENSMAC: A computational marker and cell method for free surface flows in general domains. *J. Comp. Phys.*, 110:171–186, 1994.

[47] S. Turek and M. Schäfer. Benchmark computations of laminar flow around a cylinder. In E. H. Hirschel, editor, *Flow Simulation with High-Performance Computers II*, number 52 in NNFM. Vieweg, 1996.

[48] J. Vierendeels. Implicit coupling of partitioned fluid-structure interaction solvers using reduced-order models. In H.-J. Bungartz and M. Schäfer, editors, *Fluid-Structure Interaction*, number 53 in LNCSE, pages 1–18. Springer, 2006.

[49] T. Wagner. Randbehandlung höherer Ordnung für ein cache-optimales Finite-Element-Verfahren auf kartesischen Gittern. Diploma thesis, Institut für Informatik, TU München, 2005.

[50] T. Weinzierl. Eine cache-optimale Implementierung eines Navier-Stokes Lösers unter besonderer Berücksichtigung physikalischer Erhaltungssätze. Diploma thesis, Institut für Informatik, TU München, 2005.

[51] S. Yigit, M. Heck, D. C. Sternel, and M. Schäfer. Efficiency of fluid-structure interaction simulations with adaptive underrelaxation and multigrid acceleration. *Int. J. Multiphysics*, 1:85–99, 2007.

[52] G. Zumbusch. Adaptive parallel multilevel methods for partial differential equations. Habilitationsschrift, Universität Bonn, 2001.